Refinement to Griffin Core Model and Model Mapping for Architectural Knowledge Sharing

Peng Liang, Anton Jansen, Paris Avgeriou

August, 2007
RUG-SEARCH-07-L01

Software Engineering and Architecture (SEARCH) Group
Department of Mathematics and Computing Science
University of Groningen
PO Box 800, 9700 AV Groningen
The Netherlands

Abstract

This technical report is trying to clarify the similar (overlapping and interweaving) concepts in documenting Architectural Knowledge (AK), and we argue that UML class diagram is appropriate for the representation of core model of AK for the purpose of AK sharing. A refined AK core model represented in UML is proposed, and four terminological frameworks from literatures and one domain model for industrial case for AK documentation are analyzed, and their respective concept mappings to the refined core model are presented.

This research has been partially sponsored by the Dutch Joint Academic and Commercial Quality Research & Development (Jacquard) program on Software Engineering Research via contract 638.001.406 GRIFFIN: a GRId For inFormatIoN about architectural knowledge.

Content

1 Introduction

The research work in this report is based considerably the Griffin Core Model constructed by Remco et al. in [2] using object-interaction model. But in [2], the detailed and complete mapping relationship between Griffin Core Model and shell model is not presented, and some relationships between the elements in core model are not well defined. This can hamper the effective implementation of AK grid, and further development in AK sharing and management.

In the perspective of knowledge engineering, domain model is also a kind of domain ontology model, and the Griffin core model represented in a object-interaction model is quite difficult to be mapped into the other shell models which are normally (or can be) represented in UML class diagram. For the purpose of smooth AK sharing and management, we argue that UML class diagram is more appropriate for the representation of Griffin core model with motivational argument in the next chapter. Chapter 3 presents the refined Griffin core model with the transformation rules from UML to OWL syntax. The cases of concept mapping are specified in details in Chapter 4, and Chapter 5 concludes this report.

2 Motivation

Ontology is a promising technology for knowledge representation, management and sharing, and has been widely used in some emerging fields, such as semantic web, with demonstrative specifications, e.g. RDF, OWL, etc. As Gruber defined in [1], "An ontology is a specification of a conceptualization", and in the knowledge engineering domain, ontology can be regarded as a conceptual model, which act as the central point to bridge the diverse knowledge resource, including different concepts and the instance of concept.

UML, as a standard modeling language for software-intensive systems originally, is appropriate in natural for the ontology modeling. Arguments as follows:

- UML has been the de-facto standard modeling language in software engineering with wide acceptance and tool support;
- There exists a direct relation between the class diagrams of UML and the parts of an ontology (e.g. classes, hierarchies, class attributes and axioms by OCL).

And why UML is better for the representation of AK core model? Advantages:

- Most of stakeholder of AK is professionals of software engineering, and UML could be the best language for the common understanding;
- automatic and transparent transformation from UML class diagram to OWL, more appropriate for the forward and reverse transformation from UML to OWL, and vice verse in case of change of core model, and concept mapping rules.

3 Refined Griffin Core Model

3.1 Concept and relationship clarification

The refined Griffin core model in UML as shown in Figure 1 can be regarded as a model mapping based on Griffin core model in [2] from UML perspective. We inherit all the 12 entity concepts in the Griffin core model represented in object-interaction model [2], and use some of the actions as the relationship between concepts[1], and rename some action as simple as possible for easy understanding. For example, the "perform" action between Role and Activity entity is mapped as a "perform" relationship between Role and Activity concept in the refined Griffin core model in UML.

3.2 Relationship difference

The only semantic difference between Griffin core model in [2] and the Griffin core model in UML is described below:

[1] in the research of [5], the relationship between entity can also be regarded as a concept as well, for instance, the "enforce upon" relationship in Figure 1 can be regarded as "design strength" concept.

Griffin core model in [2] does not allow multiple Stakeholders sharing the same concern. The refined Griffin core model in UML allows the AK to document this circumstance.

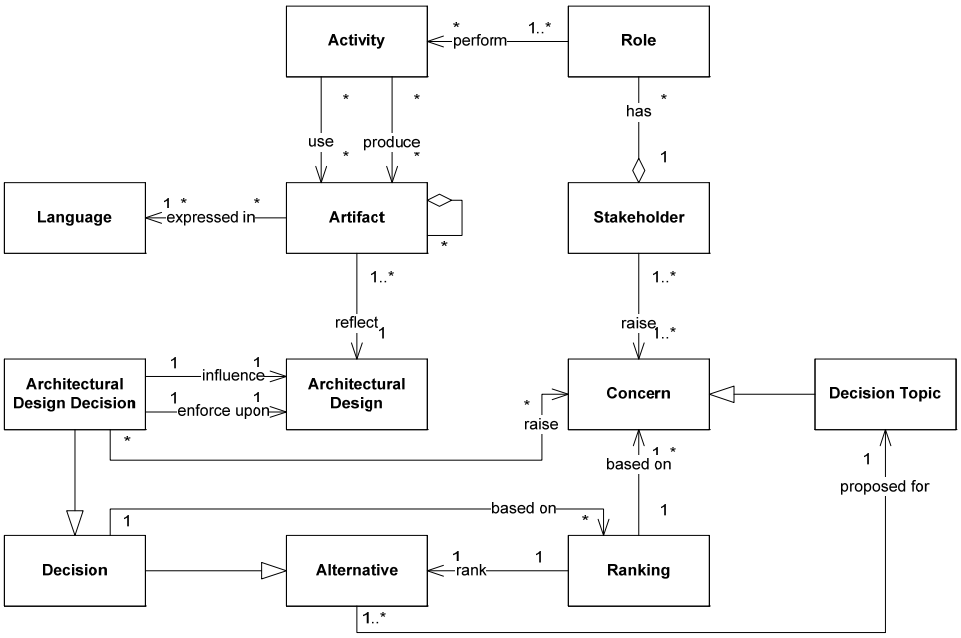

Figure 1. Refined Griffin Core Model represented in UML

3.3 Transformation from UML model to OWL model

Much of research work has been done in the filed of UML representation for ontology, such as ODM [6], and OWL as a syntax to represent ontology for the purpose of web engineering has been paid much attention in recent years. We try not to get in deep mapping rules and metamodel for the transformation between UML construct to OWL representation, but we use the simple and declarative relationship to perform the transformation as specified as follows

3.3.1 subclass relationship

The `subclass` relationship (e.g. Decision is `subclass` of Alternative) in UML can be represented by `rdfs:subClassOf` relationship in OWL directly. Sample is shown as follows:

```
<owl:Class rdf:ID="Decision">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Alternative"/>
  </rdfs:subClassOf>
</owl:Class>
```

3.3.2 Other relationships:

other relationships (e.g. Alternative is `proposed for` Decision Topic) in UML can be represented by `owl:ObjectProperty` construct in OWL, and for the purpose of mutual traceability, the backward relationship is also created using the `owl:inverseOf` construct in OWL. Sample is shown as follows:

```
<owl:ObjectProperty rdf:about="#proposed_for">
  <owl:inverseOf rdf:resource="#addressed_by"/>
  <rdfs:domain rdf:resource="#Alternative"/>
  <rdfs:range rdf:resource="#Decision_Topic"/>
</owl:ObjectProperty>
```

The visualization of OWL representation of Refined Griffin core model is shown in Figure 2 by the Jambalaya plug-in in Protégé. The box represents the class, and the arc represents the relationship between classes, including `subClassOf`, `isPartOf`, and other relationships.

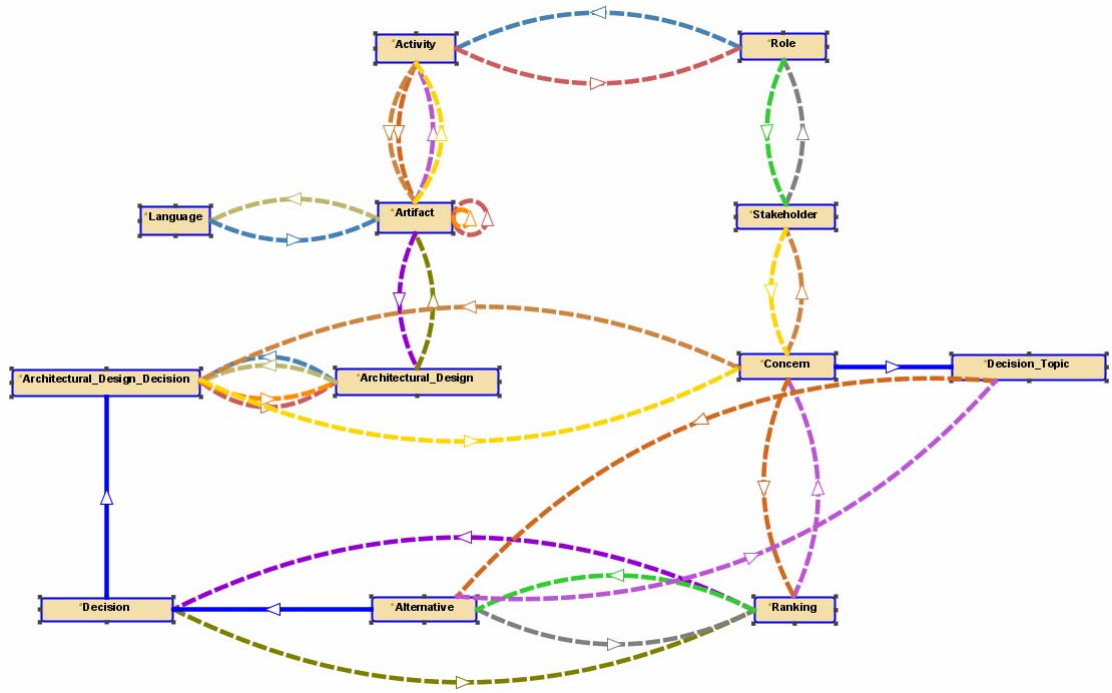

Figure 2. Refined Griffin Core Model represented in OWL shown in Protégé Jambalaya Visualization Plug-in

4 Cases of Concept Mapping

The concept mapping between refined Griffin core model in UML and the domain model from industrial cases are presented as follows respectively. The mapping relationship (e.g. `sameAs`, `subClassOf`, etc) between the concepts is defined based on the understanding of the concepts in their respective terminological frameworks, and all of concept mapping are defined manfully with the knowledge of domain expert. The concepts in candidate model (the model to be mapped onto the refined Griffin core model) are in *italic* font in the mapping table. The exceptional concept mappings refer that the concept in candidate model is mapped to the concept in the candidate model itself, and the instance mapping of AK entity based on the exceptional concept mappings will remain the original structure for further reference and usage. For the situation of concept mapping from 1 to n (e.g. *conceptA* is mapped to conceptB, conceptC and conceptD), the instance of *conceptA* should be mapped onto the instance of conceptB, conceptC and conceptD respectively by domain experts. The concepts in refined Griffin core model without mapping pair are marked in gray background.

4.1 AREL model

AREL conceptual model proposed in [7] for design traceability and reasoning, and the concept mapping between IEEE 1471-2000 standard and refined Griffin core model is specified in Table 1. The exceptional concept mappings are: *Alternative Architecture Rationale* is `compositionOf` Alternative, Ranking, Concern, *Alternative Design*, and *Alternative Behavior*, *Alternative Design* is `sameAs` *Alternative Design*, *Alternative Behavior* is `sameAs` *Alternative Behavior*.

Table 1 Concept mapping between AREL model and Refined Griffin Core Model

AREL model Concepts	**`relationship`**	**Refined Griffin Core Model**
Decision	`sameAs`	Decision
Motivational Reason	`sameAs`	*Architectural Driver*
Architectural Driver	`subClassOf`	Concern
Business Viewpoint	`subClassOf`	Language
Functional Requirement	`subClassOf`	Concern
Non-Functional Requirement	`subClassOf`	Concern
Business Environment	`subClassOf`	Concern

Information System Environment	`subClassOf`	Concern
Technology Environment	`subClassOf`	Concern
Design Outcome	`sameAs`	*Architectural Design*
Architectural Design	`sameAs`	Architectural Design
Data Viewpoint	`subClassOf`	Language
Application Viewpoint	`subClassOf`	Language
Technology Viewpoint	`subClassOf`	Language
Data Model	`subClassOf`	Artifact
Application Model	`subClassOf`	Artifact
Technology Model	`subClassOf`	Artifact
Architectural Rationale	`compositionOf`	Architectural Design Decision, Decision Topic, Alternative, Ranking
Alternative Architecture Rationale	`compositionOf`	Alternative, Ranking, Concern, *Alternative Design*, *Alternative Behavior*
Qualitative Rationale	`compositionOf`	Ranking, Concern
Quantitative Rationale	`compositionOf`	Ranking, Concern
Alternative Design	`sameAs`	*Alternative Design*
Alternative Behavior	`sameAs`	*Alternative Behavior*
		Activity
		Role
		Stakeholder

Kruchten's ontology:
Kruchten's ontology proposed in [8] for documenting mainly Architectural Design Decision, and the concept mapping between Kruchten's ontology and refined Griffin core model is specified in Table 2. The exceptional concept mappings are: *Structural Decision*, *Behavioral Decision*, and *Ban Decision* are `subClassOf` *Existence Decision*, *Constraint*, *Design Rule*, and *Guideline* are `subClassOf` *Property Decision*, *Organization*, *Process*, *Technology*, and *Tool* are `subClassOf` *Executive Decision*. The *Design Decision* is `sameAs` Architectural Design Decision or Alternative based on the value of *State*. The concept of *Risk*, *Requirement*, *Plan*, and *Design Element* are not the concepts from Kruchten's ontology, but the concepts traceable from Kruchten's ontology, and we map them onto the concepts in the refined Griffin core model as well.

Table 2 Concept mapping between Kruchten's ontology and Refined Griffin Core Model

Kruchten's ontology	**`relationship`**	**Refined Griffin Core Model**
Design Decision (State)	`superClassOf`	Architectural Design Decision, Alternative
Existence Decision	`subClassOf`	Architectural Design Decision
Property Decision	`subClassOf`	Architectural Design Decision
Executive Decision	`subClassOf`	Architectural Design Decision
Structural Decision	`subClassOf`	*Existence Decision*
Behavioral Decision	`subClassOf`	*Existence Decision*
Ban Decision	`subClassOf`	*Existence Decision*
Constraint	`subClassOf`	*Property Decision*
Design Rule	`subClassOf`	*Property Decision*
Guideline	`subClassOf`	*Property Decision*
Organization	`subClassOf`	*Executive Decision*
Process	`subClassOf`	*Executive Decision*
Technology	`subClassOf`	*Executive Decision*
Tool	`subClassOf`	*Executive Decision*
Scope	`sameAs`	Decision Topic
Rationale	`compositionOf`	Decision Topic, Architectural Design Decision, Alternative, Ranking
Cost	`subClassOf`	Ranking
Risk	`subClassOf`	Concern

Requirement	subClassOf	Concern
Defect	subClassOf	Concern
Plan	subClassOf	Concern
Design Element	subClassOf	Artifact
Implementation Element	subClassOf	Artifact
		Decision
		Stakeholder
		Language
		Role
		Activity
		Architecture Design

4.2 Tyree's template

Tyree's template proposed in [9] for Architecture Decisions documentation, and the concept mapping between Tyree's template and refined Griffin core model is specified in Table 3. The exceptional concept mappings are: *Note* is `sameAs` *Note*. Some concept in Tyree's template, such as *Related Decision*, *Related Requirement*, *Related Artifact*, and *Related Principle*, have no corresponding concepts in the refined Griffin core model, and will remain the original relationship information for further reference and usage.

Table 3 Concept mapping between Tyree's template and Refined Griffin Core Model

Tyree's template	**`relationship`**	**Refined Griffin Core Model**
Issue	subClassOf	Concern
Decision (Status)	sameAs	Architectural Design Decision, Alternative
Group	sameAs	Decision Topic
Assumption	subClassOf	Decision
Constraint	subClassOf	Concern
Position	sameAs	Alternative
Argument	compositionOf	Concern, Ranking
Implication	compositionOf	Architectural Design Decision, Concern
Note	sameAs	*Note*
		Activity
		Role
		Language
		Artifact
		Stakeholder
		Architectural Design

4.3 IEEE 1471-2000 Standard

IEEE 1417-2000 standard [3] provides a conceptual model for Architectural Description, and the concept mapping between IEEE 1471-2000 standard and refined Griffin core model is specified in Table 4. The exceptional concept mappings are *View* is `compositionOf` *Architectural Model*, and *Library Viewpoint* is `SourceOf` *Viewpoint*.

Table 4 Concept mapping between IEEE 1471-2000 standard and Refined Griffin Core Model

IEEE 1471-2000 Concepts	**`relationship`**	**Refined Griffin Core Model**
Mission	subclassOf	Concern
Environment	subclassOf	Concern
System	compositionOf	Architectural Design, Artifact
Architecture	sameAs	Architectural Design
Stakeholder	sameAs	Stakeholder
Architectural Description	subclassOf	Artifact
Rationale	compositionOf	Decision Topic, Decision, Alternative, Ranking
Concern	sameAs	Concern

Viewpoint	`subclassOf`	Language
View	`compositionOf`	*Architecture Model*
Library Viewpoint	`compositionOf`	*Viewpoint*
Architecture Model	`subclassOf`	Artifact
		Architectural Design Decision
		Activity
		Role

4.4 Astron Domain Model

The domain model proposed for the AK documentation for the Astron, whose projects are due to the long development of more than 10 years, and architectural decisions need to be shared and used over 25 years. The concept mapping between Astron domain model and refined Griffin core model is specified in Table 5. There is no exceptional concept mappings.

Table 5 Concept mapping between Astron Domain Model and Refined Griffin Core Model

Astron Domain Model	**`relationship`**	**Refined Griffin Core Model**
Author	`subclassOf`	Stakeholder
Artifact	`sameAs`	Artifact
Artifact Fragment	`partOf`	Artifact
Concern	`sameAs`	Concern
Requirement	`subclassOf`	Concern
Risk	`subclassOf`	Concern
Decision Topic	`sameAs`	Decision Topic
Alternative	`sameAs`	Alternative
Decision	`sameAs`	Architectural Design Decision, Decision
Quick Decision	`subclassOf`	Architectural Design Decision, Decision
Specification	`subclassOf`	Architectural Design Decision
		Activity
		Role
		Language
		Architectural Design
		Ranking

5 Conclusion and Future Work

Based on the concepts mapping result from different terminology frameworks and industrial domain model on AK to the refined Griffin core model, we get clearer understanding about how concept mapping on AK works and this work can be a base for the evaluation of instance mapping quality on AK.

Future work can be classified in two levels, model level and instance level.

Model level: (1) model mapping is the first step for the instance AK sharing and management, the prediction metrics and rules should be defined for the instance mapping quality evaluation, and (2) when new AK domain model comes into play for the AK sharing and management activity, how to minimize the change impact of core model modification, and the model mapping between core model and the other domain model.

Instance level: the quality of model mapping can only be evaluated practically in the instance level, i.e. for the repository of AK documentation. (1) How to define the mapping quality evaluation process in instance level, and (2) what is implicated relationship between the quality prediction in the model level and practical mapping result in instance level should be future investigated.

6 Reference

[1] T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993.

[2] Remco C. de Boer, Rik Farenhorst, Patricia Lago, Hans van Vliet, Viktor Clerc, and A. Jansen, Architectural Knowledge: Getting to the Core, in QoSA (3rd International Conference on the Quality of Software Architectures), 2007.
[3] IEEE. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Standard 1471-2000, IEEE, 2000.
[4] Object Management Group. Software Process Engineering Metamodel Specification. Technical Report formal/05-01-06, Object Management Group, January 2005.
[5] Antony Tang, Muhammad Ali Babar, Ian Gorton, and J. Han, A survey of architecture design rationale, Journal of Systems and Software, vol. 79, pp. 1792-1804, 2006.
[6] Object Management Group. Ontology Definition Metamodel Specification. Technical Report formal/06-05-01, Object Management Group, May 2006.
[7] Antony Tang, Yan Jin, and Jun Han, A Rationale-based Architecture Model for Design Traceability and Reasoning, Journal of Systems and Software, vol. 80, pp. 918-934, 2007.
[8] P. Kruchten, An ontology of architectural design decisions in software intensive systems, SVM (2nd Groningen Workshop on Software Variability Management), pp. 54-61, 2004.
[9] Tyree J and Akerman A, Architecture Decisions: Demystifying Architecture, Software, IEEE, vol. 22, pp. 19-27, 2005.